R.L. LondonによるR. Floydのソート・プログラムの正当性の証明について (プログラムの基礎理論)

AUTHOR(S):

筧, 捷彦

CITATION:

筧, 捷彦. R.L. LondonによるR. Floydのソート・プログラムの正当性の証明について (プログラムの基礎理論). 数理解析研究所講究録 1971, 119: 28-48

ISSUE DATE:

1971-07

URL:

http://hdl.handle.net/2433/106466

数理解析研究所講究録
119巻 1971年 28-48

R.L. London による R. Floyd の ソート・プログラムの 正当性の証明について

東京大学工学部　筧 捷彦

0. 序

J. McCarthy が［9］で述べているように，MTC（Mathematical Theory of Computation）の実用的な面での応用は主として，次のことにある。

a. プログラム作成時に，正しいものを作り出すための種々の道具立てを提供する．

b. プログラムについてのデバッギングの手間を省くこと．

ここでは，b. の応用部門について，実用的なプログラムに関して正当性の証明を R.L. London が与えた［8］例につきその方法を紹介し，あわせて正当性の証明に関するいくつかの試みに触れることにする。

1. Tree Sort 3

証明の対象となるのは，CACM の Algorithms のページに掲

載された，R.W. Floyd による Tree Sort 3 という名のプログラム［5］である。これは，binary search を用いたソートの変形であり，ソートの対象となるデータの入った配列以外には特別の記憶域を用いずにソートを行うものである。

プログラムは siftup(i_0, i_n) という手続きを用意して構成され，以下のようなものである。

```
0) integer i;                                              P0
1) for i := n ÷ 2 step -1 until 2 do                       P1
2) siftup(i, n) comment P1' ;                              P2
3) for i := n step -1 until 2 do
4) begin                                                   P3
5)     siftup(1, i);                                       P4
6)     exchange(M[1], M[i]);                               P5
7) end;                                                    P6
```

図1.

ここに，M (M[1], M[2], ···, M[n]) はデータの配列，nはデータの個数であり，exchange(a, b) はaとbの内容を入れ替える手続きとする。

siftup(i_0, i_n) は，

A(i) を，　M[k÷2] ≥ M[k], for $2i \leq k \leq i_n$ ―― <1>

なる不等式全部を表わすものとして，

<in>　A(i_0+1)　and　$1 \leq i_0 \leq i_n \leq n$

なる条件下に呼出されたとき，

<out> 1. $A(i_0)$

2. $M[i_0] \sim M[i_n]$ は置換される

3. $M[i_0] \sim M[i_n]$ 以外の部分は不変である

なる条件を満たすように配列Mの内容を変化させる手続きである。i_0, i_n の値が siftup により変えられることはない。

平たく言うと，

(1),(2) により，A(2) が成立するよう準備する．

(3)~(7) のループでは，

(5)の結果 M[1] が，M[1]~M[i] の中で最大となるから

(6)によって これを M[i] へ入れる。

ことによりソートを完了する。

2. 証明の方法

■ 図1. のプログラムの各行に対して，このプログラムを動作させたとき，その行に実行の流れが達する場合に成立すると考えられる命題 p0, p1, …, p6 を書き下す。([ ] では， comment の形でプログラムに書き込まれている)

■ 次に，各命題の成立を，プログラムの流れ及び意味を考えながら，ひとつひとつ証明する。

■ 以上の帰結として，各命題は，このプログラムの実行中い

かなる場合にも，各行において成立しているものとする。従ってまた，このプログラムが終止した時には，命題 p6 が成立しており，p6 が「Mはソートされた」ことを含んでいるから，このプログラムは正当（正しい）だと結論する．

London による p0～p6 及びその証明は下のようである．

p0: $A(n \div 2+1)$

p1: 1. $A(i+1)$　　2. $\langle in \rangle$ を満たす（$i_0=i$, $i_n=n$）

p1': $A(i)$

p2: 1. $M[p] \le M[p+1]$, for $n+1 \le p \le n-1$

2. $A(2)$ i.e. $M[k \div 2] \ge M[k]$, for $4 \le k \le n$

p3: 1. $M[p] \le M[p+1]$, for $i+1 \le p \le n-1$

2. $M[k \div 2] \ge M[k]$, for $4 \le k \le i$

3. $M[i+1] \ge M[r]$, for $1 \le r \le i$

4. $\langle in \rangle$ を満たす（$i_0=1$, $i_n=i$）

p4: 1. $M[p] \le M[p+1]$, for $i+1 \le p \le n-1$

2. $M[k \div 2] \ge M[k]$, for $2 \le k \le i$

3. $M[1] \ge M[r]$, for $2 \le r \le i$

4. $M[i+1] \ge M[1]$

p5: 1. $M[i] \ge M[r]$, for $1 \le r \le i-1$

2. $M[p] \le M[p+1]$, for $i \le p \le n-1$

3. $M[k \div 2] \ge M[k]$, for $4 \le k \le i-1$

p6: 1. $M[p] \leq M[p+1]$, for $2 \leq p \leq n-1$

2. $M[2] \geq M[1]$

3. $M[p] \leq M[p+1]$, for $i \leq p \leq n-1$ } ソート完了

4. M は最初の配列を置換しただけ }

証明

p0: <1>の定義から空文，従って成立

p1: 1. 第1回目は $i = n \div 2$ 及び p0 による，その他の場合は p1' 及び 1) により $i := i+1$ となっているから．

2. $i = i_0 = 2, 3, \cdots, n \div 2 < n$

p1': p1:2. により <in> が成立，従って <out> が siftup の定義により成立する．

p2: 1. 空文

2. $n \geq 4$ なら 2) が実行され，p1' 及び $i=2$ (2)) による．

$n \leq 3$ なら 空文

p3: 1. p2:1.（1回目），p5:2 及び 3) で $i := i-1$（ナルメタ）

2. p2:2.（1回目），p5:3 及び 3) で $i := i-1$（ナルメタ）

3. $i=n$ ゆえ空文(1回目)，p5:1 及び 3) で $i := i-1$（ナルメタ）

4. 5) と p3:2 による．i.e. A(2) for M[1:i]

p4: 1. p3:1 及び <out>3

2. p3:2 及び <out>1

3. p4:2 から，$k=2$ のとき $M[k \div 2] = M[1]$ 及び $\geq$ の性質

4. $i=n$ なら空文，$i\neq n$ なら，p3:3 と <out>2 によってここでの M[1] は，p3成立時のある r, $1\leq r\leq i$ に対して M[r] に等しいことによる

p5: 1. p4:3 及び 6)による

2. p4:1 及び 6)による

3. p4:2 及び 6)による

p6: $n\geq 2$ なら 3)~7) が実行されて，

1. は p5:2 及び $i=2$，2. は p5:1 及び $i=2$

3. は p6:1, 2 による．

$n\leq 1$ なら 1.~3. は空文．

4. は，Mに関する操作が，sift up 及び exchange によってのみ行われ，かつ，M内の置換に限られるという仮定による（p0~p5 に各々 p6:4 を付加する）

■このプログラムが終止することを証明する．sift up は call by value によるから，sift up, exchange が共に終了するものなら，終止は明らか．exchange が終了することも明らか故．sift up が終止することのみ証明すればよい。

[sift up のプログラム，各証明は，第4章に付録としてつけてある．]

3. 正当性証明の試み

今迄にプログラムが正当であることを証明し，あるいは保障するために用いられて来た手段としては，デバッギングがある。実際，この Tree Sort 3 に対しても，P.S. Abrams によって バローズ B-5500 を用いてデバッグが行われ，データ数 50～1000（50刻み）のものにつき，各50 種類のランダムなデータを作り出して試験したところ正しく動作した，との報告がある［ 1 ］.

しかしながら，デバッギングによる方法は，実際にデータとして与えたものに対して正しく動作すること以上には保障がない。例えば，P.S. Abrams のデバッギングの結果に対しては，R.L. London が述べているとおり，データ数が奇数の時も本当に正しく動作するのか，という疑問が残る。

これに対して，テバッギングによらずにプログラムの正当性を言おうとする試みがいくつか行われてきた.

3.1 P. Naur は［ 14 ］で，正しいプログラムを作り上げるという観点からではあるが，次の様な証明方法を提唱している。

a. プログラムの各ステップに対して comment の形で，そのステップに入る前及び実行後に成立している（はずの）条件を，数学的記法もしくはそれに近い形で付け加

える．（彼はこれを general snapshot と呼んだ）

b．各ステップに対して，その直前に実行される可能性のあるステップの実行後の条件から，そのステップに入る前の条件が出てくること，及び，入る前の条件下に，そのステップを実行した時に，実行後の条件が成立することを確める．

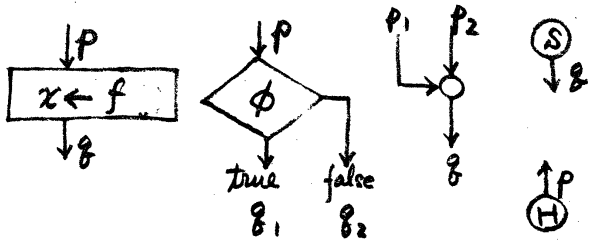
彼は，最小値を求める ALGOL 60 のプログラムに対して，この方法により正しいこと（ないし，正しいプログラムを作りあげること）を述べている。この場合，

c．証明の過程で，入る前の条件，出る時の条件がことごとく尽くされていることを確める

ように注意を与えている。

彼のこの方法は，我々がプログラムを作る場合に採用している方法であると言うことができる。R.L. London の方法もこの方法に従っている．

3.2 R.W. Floyd は［4］で，この P. Naur の方法をより一層形式化している。彼は，下図の5種類（代入，判定，合流，出発点，終了点）の command から成る flow-chart language を定義し，これによ



り作られるプログラムについての正当性の証明を次のように与えた.

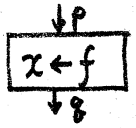
a. 各 command の入口, 出口の矢印に (そこで成立していると考えられる) 命題 p, g を割当てる.

b. 各 command につき, その入口, 出口の命題 p, g から, その command の種類によって定まった一定の規則 (verification condition* と呼ぶ) により作り出される命題の成立を証明する.

c. すべての command に対して b. の証明を終えたとき, プログラムは, 出発点での条件 g の下で, 終止点での条件 p を満たすように動作すると結論する.

更に進んで, 彼は, verification condition が満たしていなければならない性質を公理として掲げ, flow-chart language に限らず, この公理をみたすような verification condition を各々の文(の形) に与えることが即ち language の意味を与えることになると述べている.

---

* 代入  に関する verification condition は

$$Vc_{x\leftarrow f}(p,g) : (\exists x_0)(x=f(x_0,y)\wedge R(x_0,y)) \supset g .$$

但し, p は $R(x,y)$ の形, y は x 以外の変数を示す.

プログラムの終止性についても触れて，各命題の割当てと同時に，プログラム上の変数から，順序集合への写像 w を各々割当て，各 command をへる毎に，w の値が小さくなることが言えれば，終止すると結論できると述べている．この方法は，R.L. London の証明においても（部分的に）用いられている．

3.3 D.E. Knuth も［7］に於いて，アルゴリズの証明を与えるために，同様の方法を採用している．彼は，数学的帰納法を，アルゴリズの正しさの証明の一般的方法の基本としてとり，次のように述べている．

a. アルゴリズを流れ図に表現し，各矢印に，そこで成立している（と思われる）条件を名札としてつける．

b. 流れ図中の各演算の箱に対して，その箱へ達している矢印の名札の条件の内どれか1つが成立しているなら，演算の結果，箱から出ている矢印の名札の条件がすべて成立することを証明する．

c. b. がすべての箱につき証明できれば，a. でつけたすべての条件は，アルゴリズの実行中に成立していると結論する．

Knuth は Floyd の方法の内，特に"局所的な証明の結果，全体の証明が得られる"という，帰納的な面のみを採り上げて，アルゴリズの証明に用いたといえる．

3.4 Z. Manna は[12]において，R. Floyd の flow-chart language と同等のものに対し，1つのプログラムに，ある命題を対応させる方法を与え，この命題の（1階述語論理での）非充足性（unsatisfiability）が，プログラムの終止性と等価であることを示した．更に，正当性が，終止性の証明に含まれることを示している。彼は更に[13]において，同様のことを，1つの再帰的定義による定まるプログラムに対しても行っている．

C.A.R. Hoare も，ALGOL-like な language に対して，正当性の証明のために，その公理と，推論規則とを与えている。例えば 代入文についての公理は，（一般に，文Qにつき，初め条件Pが成立しているとき，実行後Rが成立することを $P\{Q\}R$ であらわして）下によって与えられる．

$\vdash P_0\{x:=f\}P$ ， P_0 は P 中のすべての x に f を代入して得られるもの

彼はまた，こうした公理的接近によって，language の定義において無定義とされる様な場合についても（例えば，整数値が大きくなりすぎた場合）明確に取扱えるのではないか，との意見を述べている。[6]

その他，R.M. Burstall による structual induction を用いた正当性の証明[2]（structual induction は，

J. McCarthy, J.A. Painter によるコンパイラの正当性証明の試み[11]に於いても用いられており, J. McCarthy が用いた recursion induction [10]の特別な場合といえる)や, 同じく, P.J. Landin との共著による代数の準同型写像を用いた方法[3]等もある. 以上。

4. 付

siftup に対する 正当性及び終止性の証明

- $A(i)$ は本文中の定義どおり.
- i_0, M_0 は各々 siftup が呼ばれた時の i, M の値とする.
- $M = p(M_0)$ は, M が, M_0 の置換により得られることを示す.
- $M = p(M_0)$ with $M[k] := copy$ は, $M[k] := copy$ を行えば $M = p(M_0)$ であることを示す.

procedure siftup(i, m); **value** i, m; **integer** i, m;

begin **real** copy; **integer** j;

1.1	$1 \le i_0 = i \le m \le n$ (array size)
1.2	$A(i_0+1)$
1.3	$M = p(M_0)$

2) copy := M[i];

3) loop: j := 2 × i;

3.1	$i \le m$
3.2	$2i = j$
3.3	$i = i_0$ or $i \ge 2i_0$
3.4	$M = p(M_0)$ with $M[i] := copy$
3.5	$A(i_0)$ or ($i = i_0$ and $A(i_0+1)$)
3.6	$M[i \div 2] > copy$ or $i = i_0$
3.7	$M[i \div 2] \ge M[i]$ or $i = i_0$

4) **if** $j \le m$ **then**

5) **begin** **if** $j < m$ **then**

6) **begin** **if** M[j+1] > M[j] **then** j := j+1 **end**;

6.1	$i = j \div 2$
6.2	$2i \le j \le m$
6.3	$i = i_0$ or $i \ge 2i_0$
6.4	$M = p(M_0)$ with $M[i] := copy$

6.5	$A(i_0)$ or ($i=i_0$ and $A(i_0+1)$)
6.6	$M[i\div 2] > copy$ or $i=i_0$
6.7	$M[i\div 2] \geq M[i]$ or $i=i_0$
6.8	($2i<n$ and $M[j]=max(M[2i],M[2i+1])$ or ($2i=n$ and $M[j]=M[n]$)
6.9	$M[i] \geq M[j]$ or $i=i_0$

7) **if** $M[i] > copy$ **then**

8a) **begin** $M[i] := M[j]$;

8.1	$i=i_0$ or $i \geq 2i_0$
8.2	$2i \leq j \leq m$
8.3	$M[j\div 2]=M[i]=M[j] > copy$
8.4	$M[i\div 2] \geq M[j]$ or $i=i_0$
8.5	$M=p(M_0)$ with $M[j]:=copy$
8.6	$A(i_0)$

8b) $i := j$;

8.7	$i \geq 2i_0$
8.8	$i=j \leq m$
8.9	$M[i\div 2] > copy$
8.10	$M[i\div 2] \geq M[i]$
8.11	$M=p(M_0)$ with $M[i]:=copy$
8.12	$A(i_0)$

8c) **go to** loop

end

9) **end** ;

9.1	M[j] ≤ copy if reached from 7 or 2i = j > m if reached from 4

10) M[i] := copy

10.1	$M = p(M_0)$
10.2	$A(i_0)$

end siftup ;

各命題に対する証明

1.1 ~ 1.2　siftup を使う時の仮定

1.3　p は恒等置換

3.1 ~ 3.7　2) から来た時

- 3.1 は 1.1 による
- 3.2 は 3) の代入による
- 3.3, 3.5~3.7 は 1.1 により $i = i_0$, 及び 1.2
- 3.4 1.3 及び 2) の代入による

8c) から来た時

- 3.1 ~ 3.7 は順に, 8.8, 3) の代入, 8.7, 8.11, 8.12, 8.9, 8.10 による

6.1　3.2 により $j=2i$, 6) の代入があって $j=2i+1$

$\therefore\quad i=j\div 2$

6.2　4) により $j\leq m$, j への代入は 6) に限られ,

その時は 5) により $j<m$　$\therefore$　$j\leq m$ および 6.1

6.3 ~ 6.7　順に 3.3 ~ 3.7 による

6.8　4) は true. 5) が false なら 3.2 により $j=2i$

$=m$　$\therefore$ 6.8 の第2項が成立. 5) が true なら,

6) の前では, $j=2i<m$ and $M[j+1]=M[2i+1]$,

6) の結果 $j=2i$ または $j=2i+1$　$\therefore$ 6.8 の第1項成立

6.9　$i\neq i_0$ なら, 6.1 より $A(i_0)$. 6.2, 6.3 より

$2i_0\leq 2i\leq j\leq m$. $A(i_0)$ の定義と 6.1 から $M[i]=M[j\div 2]\geq M[j]$

8.1, 8.2　順に 6.3, 6.2 による

8.3　6.1 より $i=j\div 2$, 7) より $M[i]>copy$, 8a) より

$M[i]=M[j]$

8.4　6.7 および 6.9 による

8.5　6.4 と 8a) の代入により, $M[i]$ のかわりに $M[j]$ へ

copy を代入する要あり. 8.1 と 8.2 から $i_0\leq i\leq m$

ゆえ, 8a) の代入は $M[i_0]\sim M[m]$ 間に行われる

8.6　・8a) の代入により,

6.8 の第1項成立なら　$M[i]\geq M[2i], M[2i+1]$

6.8 の第2項成立なら　$M[i]=M[j]=M[m]=M[2i]$,

M[2i+1] は存在しない。

- $A(i_0) \supset A(i_0+1)$ ゆえ 6.5 では $A(i_0+1)$
- $i=i_0$ なら, $A(i_0+1)$ と M[i] の上記条件より $A(i_0)$

  $i \neq i_0$ なら 6.5 で $A(i_0)$, 8a) の代入にもかかわらず, 8.3, 8.4 から $M[i \div 2] \geq M[j] = M[i]$

  かつ, $M[i] \geq M[2i], M[2i+1]$ ∴ $A(i_0)$

8.7 8.1, 8.2 から 8b) の代入による

8.8 8.2 及び 8b) の代入による

8.9 8.3 及び 8b) の代入による

8.10 8a) の後では 8.1, 8.2 から $2i_0 \leq j \leq m$. 8.6 により $M[j \div 2] \geq M[j]$. ∴ 8b) の代入による

8.11 8.5 及び 8b) の代入による

8.12 8.6 による

9.1 9) へは 7) が false の時 (6.9 及び 7) により $M[j] \leq copy$)

. または 4) が false の時 (3.2 及び 4) により $2i = j < m$) しか来ない

10.1~10.2

- 7) から来た時

  10.1 は 6.4 と 10) の代入による. (6.2, 6.3 から $i_0 \leq i \leq m$)

  10.2 9.1, 6.2, 6.8 と 10) の代入により $M[i] = copy \geq M[j] \geq M[2i]$ かつ M[2i+1] が存在するなら $M[j] \geq M[2i+1]$.

  $i = i_0$ なら 8.6 と同様に証明可. $i \neq i_0$ のとき

6.6 と 10) の代入により $M[i \div 2] > copy = M[i]$.

6.5 の $A(i_0)$ によって 10.2 成立

・4) から来た時

10.1 は 3.4 と 10) の代入による (3.1, 3.3 から $i_0 \leq i \leq m$)

10.2 $2i > n$ (4)) は $A(i_0)$ が $M[i] \geq \cdots$ の形の不等式を含まぬことを示す. $i = i_0$ なら 3.5 により, $i \neq i_0$ なら 3.6 と 10) の代入により $M[i \div 2] > copy = M[i]$, 3.5 の $A(i_0)$ と合わせて 10.2 成立 ged

終止することの証明

・siftup のプログラムがループに入って止まらなくなる唯一の可能性は, 3) から 8c) へ至たり再び 3) へ戻る部分についてだけである.

・i の値 (8b)), j の値 (3) 及び 6)) が変えられるのは上記のループ内だけであり, かつループをまわる毎に値は変わる。

・4) での判定に従い, j の値が増加してゆくことを示せば終止の証明になる.

・1.1 により $i \geq 1$ 即ち $2i > i$.

8b) では $j = i < 2i$. 3) では $j = 2i$.

従って j (3) に於ける) $= 2i > i = j$ (8b) に於ける).

6) での代入が行われても, j の値は増加するのみ. ged

文 献

1) Abrams, P.S., Certification of ALgorithm 245, CACM, vol.8, No.7, p.445 (1965).

2) Burstall, R.M., Proving properties of programs by structual induction, the Computer Journal, vol.12, No.1, pp.41-48 (1969).

3) Burstall, R.M. and Landin, P.J., Programs and their Proofs: an Algebraic Approach, Machine Intelligence 4, Meltzer, B. and Michie, D., (Eds.), American Elsevier, New York, pp.17-43 (1969).

4) Floyd, R.W., ASSINING MEANINGS TO PROGRAMS. Proceedings of Symposia in Applied Mathematics, vol.19, Mathematical Aspects of Computer Science, pp. 19~32 (1967).

5) Floyd, R.W., Algorithm 245, TREESORT 3, CACM, vol.7, No.12, p.701 (1964).

6) Hoare, C.A.R., An Axiomatic Basis for Computer Programming, CACM, vol.12, No.10, pp.576-583 (1969).

7) Knuth, D.E., The Art of Computer Programming, Vol.1 - Fundamental Algorithms, Addison-Wesley, Reading, Mass., Sec. 1.2.1 (1968).

8) London, R.L., CERTIFICATION OF ALGORITHM 245 [M1] TREESORT 3 : PROOF OF ALGORITHMS - A NEW KIND OF CERTIFICATION, CACM, vol. 13, No. 6, pp. 371-373 (1970).

9) McCarthy, J., TOWARDS A MATHEMATICAL SCIENCE OF COMPUTATION, Information Processing 1962, Proceedings of IFIP Congress 62. Popplewell, C.M., (Ed.), North-Holland, Amsterdam, pp. 21-28 (1963).

10) McCarthy, J., A BASIS FOR A MATHEMATICAL THEORY OF COMPUTATION, Computer Programming and Formal Systems, Braffort, P. and Hirschberg, D., (Eds.), North-Holland, Amsterdam, pp. 33-70 (1963).

11) McCarthy, J. and Painter, J.A., CORRECTNESS OF A COMPILER FOR ARITHMETIC EXPRESSIONS, Proceedings of Symposia in Applied Mathematics, vol. 19, Mathematical Aspects of Computer Science, pp. 33-41 (1967).

12) Manna, Z., Properties of programs and the first-order predicate calculus, JACM, vol. 16, No. 2, pp. 244

-255 (1969).

13) Manna, Z. and Pnueli, A., Formalization of Properties of Functional Programs, JACM, vol. 17, No. 3, pp. 555-569 (1970).

14) Naur, P., PROOFS OF ALGORITHMS BY GENERAL SNAPSHOTS, BIT, vol. 6, pp. 310-316 (1966).